Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/37/20230506

Parallel programming: Driving the computational surge in AI

Yumeng Ma

University of Georgia, Georgia, 30606, The United States

ym08151@uga.edu

Abstract. With the meteoric ascent and ongoing advancements in the realm of Artificial Intelligence (AI), there's an escalating demand for potent computational capabilities. Meeting the exacting computational demands of burgeoning AI technologies has become a pivotal topic in contemporary research. This study is anchored on the theme: "Impact of Parallel Programming on AI." The objective is to delve into the potentialities and value that parallel programming technology might introduce to the AI sector. To deeply probe the integration of parallel programming in AI, we've employed a meticulous research methodology. This process disentangles and scrutinizes the prevalent parallel programming techniques and their tangible applications within the AI sphere. Such a method offers a nuanced grasp of the fusion of parallel programming with AI and the distinct advantages that ensue. Findings reveal a substantial enhancement in the efficacy of AI models when they leverage parallel programming techniques. This is especially salient in scenarios involving large-scale data training and intricate model architectures. Most notably, parallel programming dramatically slashes AI training durations. This swift training paves the way for rapid iterations and fine-tuning. In conclusion, parallel programming emerges as a game-changer in the AI domain. It doesn't merely amplify AI performance but also lays a robust technical bedrock for AI's sustained and swift evolution. This opens up a vista of novel prospects and avenues for forthcoming exploration and application in AI.

Keywords: Parallel Programming, AI, Open MPI, Big Data Set.

1. Introduction

The field of artificial intelligence has been evolving since its foundational goals were established at the Dartmouth Conference in 1956. The landmark victory of AlphaGo over the world Go champion, Lee Sedol, in 2016 marked a pivotal moment in AI's progression. Subsequent advancements, such as the advent of GPT chatbots, the public's access to self-driving cars, and the introduction of Alpha Zero, underscored that AI was entering a phase of accelerated evolution. In this rapidly shifting landscape, many AI models demand substantial computational power for faster training, effectively shortening research and development cycles. Parallel programming emerges as a solution, capitalizing on the multiple cores across one or more processors to execute numerous tasks concurrently. This approach harnesses data, task, and hardware parallelism. Not only does parallel programming expedite the training of AI models, but it also offers the capability to process data on a vast scale. Through parallel programming, tasks can be distributed and executed across multiple machines in tandem. In essence, Open MPI lays the groundwork for multitasking and managing vast datasets within the AI realm.

2. Fundamentals of Parallel Programming

2.1. Parallel Programming Patterns and Examples

Parallel programming has the following common modes: data parallelism, task parallelism, pipeline parallelism, partitioned global address space, message passing, shared memory parallelism, and combinations of parallel modes. Data parallelism refers to the process of slicing and dicing an entire piece of data into smaller pieces and then handing them off to separate tasks or threads for processing, such as vector addition, which can be used in data parallelism. Task parallelism refers to the logical division of a problem into multiple separate tasks that can be run in parallel. Ray tracing technology utilizes this model, which allows each ray to be calculated separately. Pipeline parallelism refers to the process of refining and dividing a problem so that it consists of many steps like a factory assembly line. Each step is then processed by a separate processing element. Partitioning the global address space means that the memory can be partitioned across multiple processors, but each processor can access the global address space. Message passing means that separate tasks communicate by sending and receiving messages on the display. Shared memory parallelism means that processors share a global address space and they can access the memory quickly, for example OpenMP is a multi-threaded parallel programming model for shared memory systems.

2.2. Parallel programming tools for software

Existing editing models and libraries are OpenMP, MPI, Pthreads, CUDA, OpenCL. debugging tools are GDB, TotalView, AllineaDDT. performance analysis tools are gprof, VTune, NVIDIA Nsight, TAU (Tuning and Analysis Performance Analysis Utilities)

3. Computational Demands of AI Models and Algorithms

3.1. Complexity of Deep Learning Models

Deep learning model is a model born based on artificial neural network [1], its complexity is far from the complexity of biological neural network, but because of the data it needs to deal with and the regularization problem makes it also has a certain degree of complexity. The complexity of deep learning models is mainly reflected in the model structure, the number of parameters and computational requirements, as well as regularization and generalization. In terms of model structure, for a function that can be expressed by a network structure with depth n, if we want to use a network structure with depth less than n to express this function, the computational factor may increase exponentially, which makes the computation extremely complex. Because deep learning models have a large number of parameters, they require a large amount of data for training, which also increases the complexity and computational demand of deep learning models.

3.2. The need for computing power for big data sets

Big data is a collection of many data, due to its huge amount of data it has a high demand for computing power. Data mining, data generation, data organization, storage and management, all of these require a high degree of computational power to do [2].

4. Applications of Parallel Programming in AI Training

4.1. Differences and Applications of Data Parallelism and Model Parallelism

As mentioned earlier, data parallelism is the process of dividing a data set into small subsets and processing these subsets in parallel on multiple processors or devices. Model parallelism is the process of dividing the model itself into multiple parts and processing them separately on multiple processors or devices. The only difference between them is that one slices and dices the data and the other slices and dices the model. Both data parallelism and model parallelism can be used for deep neural network learning and can be enhanced on HPC-class systems [3].

4.2. Strategies and Challenges for Distributed Training

Distributed learning in wireless mode and distributed learning in non-broadcast mode, both of which require an efficient method for distributed computation [4,5]. For a distributed training strategy one can use data parallelism and model parallelism for training [6]. However, in distributed environments, communication between the machines is often a bottleneck, which results in communication overheads and the need to synchronize the model states on the machines in order to ensure that all the data is consistent. In the case of long time-span training, a distributed system must be designed to recover from failures in order to avoid stopping the training due to an error at a node. The data load on each node must be balanced during training, as unbalanced loads can lead to wasted resources and reduced training speed. Distributed training has many challenges, the above is only a few of them, the implementation of distributed training is not simple, need to be supported by adequate technology and knowledge.

5. Influences of Parallel Programming on AI Performance

AIs are like humans, if they want to acquire a certain skill they have to go through continuous training. After acquiring the skill, if they want to enhance it, they will continue to train, just like an athlete who trains a lot every day in order to do better in the competition. So, if the AI is trained more times in the same period of time, the shorter the time it takes. Parallel programming parallelizes the code by dividing the data into multiple parts, processing them separately, and then summing up the results. This approach allows the AI to be trained in different parts at the same time, which greatly reduces the training time. However, for some models with very large parameters, it is not possible to fit all the parameters into the memory of a single GPU. In such cases, model parallelism in parallel programming can be used to distribute different parts of the model across multiple GPUs, allowing the model to be trained and inferred beyond the capacity of a single computing resource.

6. Case Studies

6.1. Examples of successful AI parallel implementations

Parallel computing has achieved good applications in several AI and deep learning projects. BERT (Bidirectional Encoder Representations from Transformers) is a large-scale training language model, BERT adopts the technology of data parallelism and model parallelism [7], and uses multi-GPU for Data parallelism has become a standard practice for training this type of model. OpenAI's GPT series uses model parallelism in the process of training its AI, because of the huge size of the model in the GPT series, it needs to cut the model and then distribute it for parallel processing. DeepSpeed by Microsoft is a library for deep learning optimization that provides a range of optimization techniques, including efficient model parallelism [8].

6.2. Case studies of challenges and problems encountered

Although it is possible to slice and dice large models using model parallelism and then process them using multiple GPUs, for some models, even if the server has multiple GPUs, the model cannot be adapted to the model because of the limited memory capacity of GPUs [9]. In the case of efficient large-scale language model training on GPU clusters using megatron-LM by Deepak Narayanan, Muhammad Shoebhai, and others, they proposed to solve this problem by using new model parallelism, tensor parallelism and pipeline parallelism. However, the unplanned use of these parallelism methods can lead to GPU scaling problems. For combining these parallelism approaches, they used the PTD-P approach, which is a method derived by combining pipeline model parallelism and tensor model parallelism with data parallelism.

6.3. Lessons learned and experiences extracted from cases

This case shows that AI parallel implementation requires in-depth consideration of multiple aspects, from algorithm and model design to hardware and communication. What people can do is to let the technology advance through continuous research and experiments, and then develop multiple strategies and tools to solve these problems and realize efficient AI parallel training.

7. Future Prospects and Trends

As far as the future of parallel programming is concerned, new parallel architectures and tools may emerge if the performance of the hardware side is improved. For example, the recent room-temperature superconductivity [10], if realized, will greatly increase the energy efficiency of computer chips. This means that more efficient parallel computing can be realized. It is expected that the future will see the use of more heterogeneous computing, combining multiple computational units such as CPUs, GPUs, TPUs and FPGAs to improve performance and energy efficiency. Parallel programming frameworks will need to support this complex computing environment and provide users with simplified tools and APIs.

8. Conclusion

Parallel programming plays an indispensable role in the realm of Artificial Intelligence (AI). By facilitating simultaneous computations, it drastically shortens training durations, paving the way for the execution of grand-scale models that would otherwise remain unrealized. Moreover, it optimizes the utilization of hardware resources, ensuring that programs run with maximal efficiency. However, as with any burgeoning technology, parallel programming isn't without its share of challenges. Notably, the high communication overhead often impedes seamless execution, as multiple processes strive to coordinate in real-time. There's also the persistent issue of load imbalance, where some computing units might be overburdened while others remain underutilized. Such disparities can lead to inefficient resource allocation and elongated computation times. Additionally, the inherent complexity of parallel programming presents steep learning curves for developers, potentially stifling innovation. As systems scale, fault recovery becomes another critical concern. A single node's failure can disrupt the entire system, leading to significant data loss or process interruption. And of course, memory constraints can act as bottlenecks, limiting the size and scope of the tasks that can be parallelized.

References

[1] Hao, Z. (2020). Big Data and Artificial Intelligence Modeling for Drug Discovery. Annual Review of Pharmacology and Toxicology, 60, 573-589.

[2] Dryden, N., Moon, T., Jacobs, S. A., & Van Essen, B. (2016). Communication quantization for data-parallel training of deep neural networks. In 2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC) (pp. 1-8). IEEE.

[3] Amiri, M. M., & Gündüz, D. (2020). Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air. IEEE Trans. Signal Process., 68, 2155-2169.

[4] Zhu, G., Wang, Y., & Huang, K. (2020). Broadband analog aggregation for low-latency federated edge learning. IEEE Trans. Wireless Commun., 19(1), 491-506.

[5] Amiri, M. M., & Gündüz, D. (2020). Federated learning over wireless fading channels. IEEE Trans. Wireless Commun., 19(5), 3546-3557.

[6] Chen, M., Gündüz, D., Huang, K., Saad, W., Bennis, M., Feljan, A. V., & Poor, H. V. (2021). Distributed learning in wireless networks: Recent progress and future challenges. IEEE Journal on Selected Areas in Communications, 39(12), 3579-3605.

[7] Alaparthi, S., & Mishra, M. (2020). Bidirectional Encoder Representations from Transformers (BERT): A sentiment analysis odyssey. arXiv preprint arXiv:2007.01127.

[8] Smith, S., Patwary, M., Norick, B., LeGresley, P., Rajbhandari, S., Casper, J., ... & Catanzaro, B. (2022). Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. Ar Xiv preprint arXiv:2201.11990.

[9] Narayanan, D., Shoeybi, M., Casper, J., LeGresley, P., Patwary, M., Korthikanti, V., ... & Zaharia, M. (2021). Efficient large-scale language model training on gpu clusters using megatron-lm.

In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (pp. 1-15).

[10] Kumar, K., Karn, N. K., & Awana, V. S. (2023). Synthesis of possible room temperature superconductor LK-99: Pb9Cu (PO4) 6O. Superconductor Science and Technology.